D Y PATIL DEEMED TO BE UNIVERSITY
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Subject Name: Design and Analysis of Algorithm

Module No:5

Module Name: Tractable and Intractable Problems

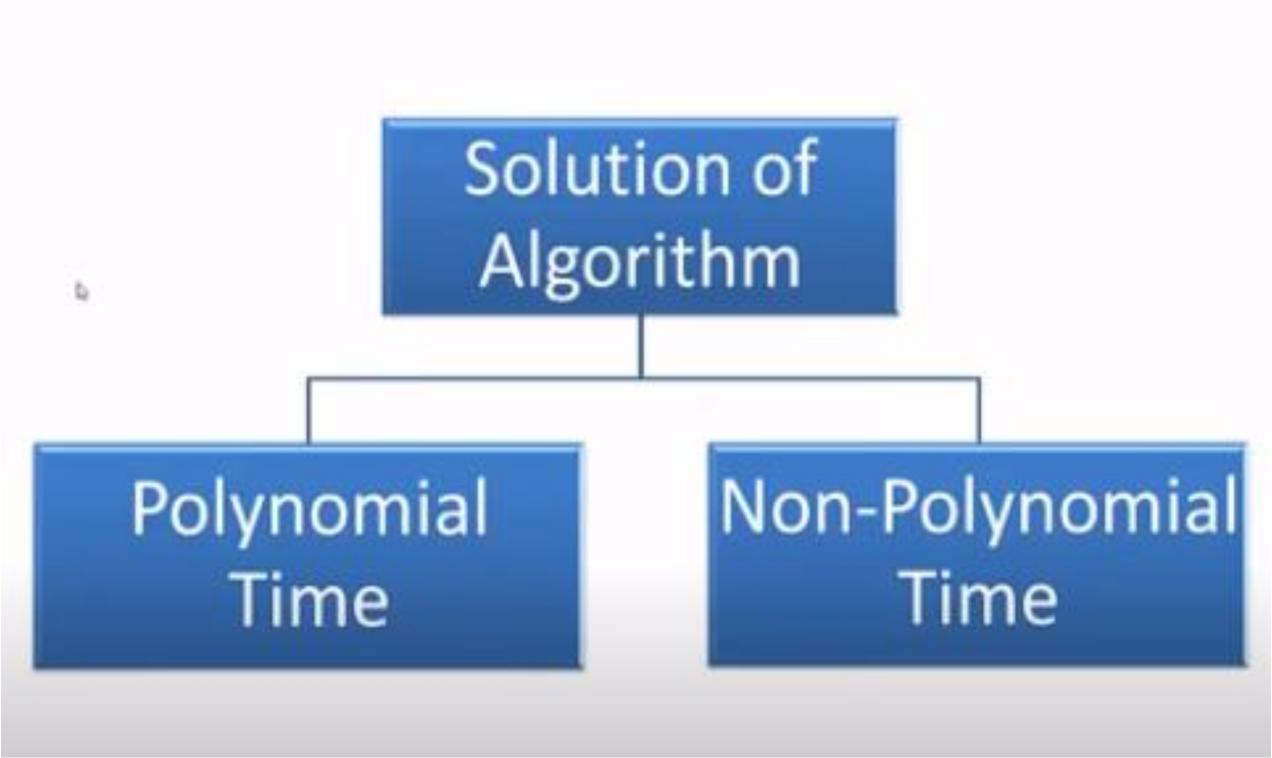
Faculty Name : Dr.Vanita Mane

Lecture 31

Basic concepts of P, NP, NP Hard and NP Complete problems

Polynomial(P) and non deterministic polynomial(NP)algorithms

P and NP



Solution of Algorithm
Polynomial Time
Non-Polynomial Time

Polynomial-time algorithm

- **A *polynomial-time algorithm*** is an algorithm whose execution time is either given by a polynomial on the size of the input, or can be bounded by such a polynomial. Problems that can be solved by a polynomial-time algorithm are called *tractable* problems.
- Sorting algorithms usually require either O(n log n) or O(n^2) time. Bubble sort takes linear time in the best case, but O(n^2) time in the average and worst cases. Heapsort takes O(n log n) time in all cases. Quicksort takes O(n log n) time on average, but O(n^2) time in the worst case.

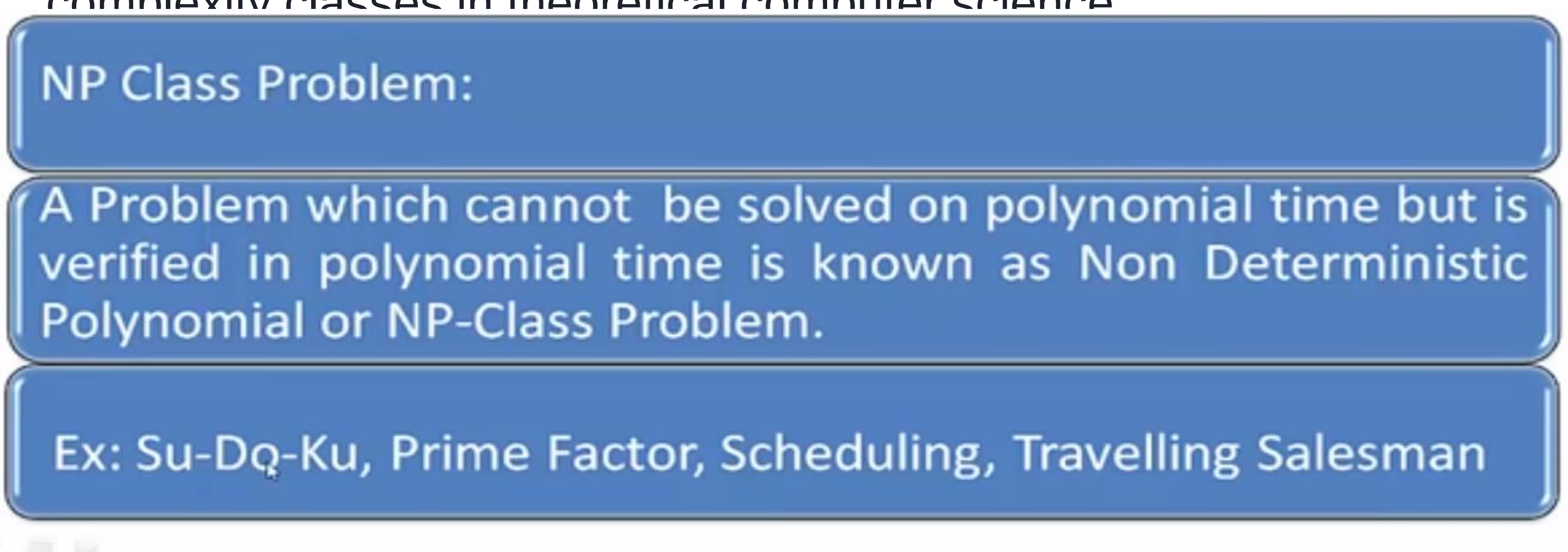
P Class Problem:

A Problem which can be solved on polynomial time is known as P-Class Problem.

Ex: All sorting and searching algorithms.

Non-deterministic Polynomial Time

- If an algorithm whose execution time is proportional to N takes a second to perform a computation involving 100 elements, an algorithm whose execution time is proportional to N^3 takes almost three hours. But an algorithm whose execution time is proportional to 2^N takes 300 quintillion years. And that discrepancy gets much, much worse the larger N grows.
- **NP, for non-deterministic polynomial time**, is one of the best-known complexity classes in theoretical computer science



Various problems in NP

Optimization Problems:

» An optimization problem is one which asks, “What is the optimal solution to problem X?”

» **Examples:**

- 0-1 Knapsack
- Fractional Knapsack
- Minimum Spanning Tree
- Decision Problems

» An decision problem is one which asks, “Is there a solution to problem X with property Y?”

» Examples:

- Does a graph G have a MST of weight ≤ W?

Various problems in NP (cont...)

- An optimization problem tries to find an optimal solution
- A decision problem tries to answer a yes/no question
- Many problems will have decision and optimization versions.

» Eg: Traveling salesman problem

- optimization: find hamiltonian cycle of minimum weight
- decision: find hamiltonian cycle of weight < k

Tractability

- In contrast, P (polynomial time) is the set of all decision problems which can be solved in polynomial time by a Turing machine.
- Roughly speaking, if a problem is in P, then it's considered **tractable**, i.e. there exists an algorithm that can solve it in a reasonable amount of time on a computer.
- If a problem is not in P, then it's said to be **intractable,** meaning that for large values it would take far too long for even the best supercomputer to solve it - in some cases, this means millions or even billions of years i.e.as they grow large, we are unable to solve them in reasonable time.

Polynomial time: $O(n^2)$, $O(n^3)$, $O(1)$, $O(n \lg n)$

Not in polynomial time: $O(2^n)$, $O(n^n)$, $O(n!)$

DY PATIL UNIVERSITY NAVI MUMBAI

Examples…

Polynomial time	Non in polynomial time
Linear Search –O(n)	0-1 Knapsack –$O(2^n)$
Binary Search –O(log n)	TSP- $O(2^n)$
Insertion Sort- $O(n^2)$	Sum of Subset- $O(2^n)$
Merger sort – O(nlogn)	Graph coloring- $O(2^n)$
Matrix Multiplication- $O(n^3)$	Hamiltonian Cycle- $O(2^n)$

Intractable Problems

- Can be classified in various categories based on their degree of difficulty, e.g.,
  - NP
  - NP-complete
  - NP-hard
- Let's define NP algorithms and NP problems …

The NP Class Problems, it is verified in polynomial time.

The P Class Problems, not only it is solved on polynomial time but it is verified also in polynomial time.

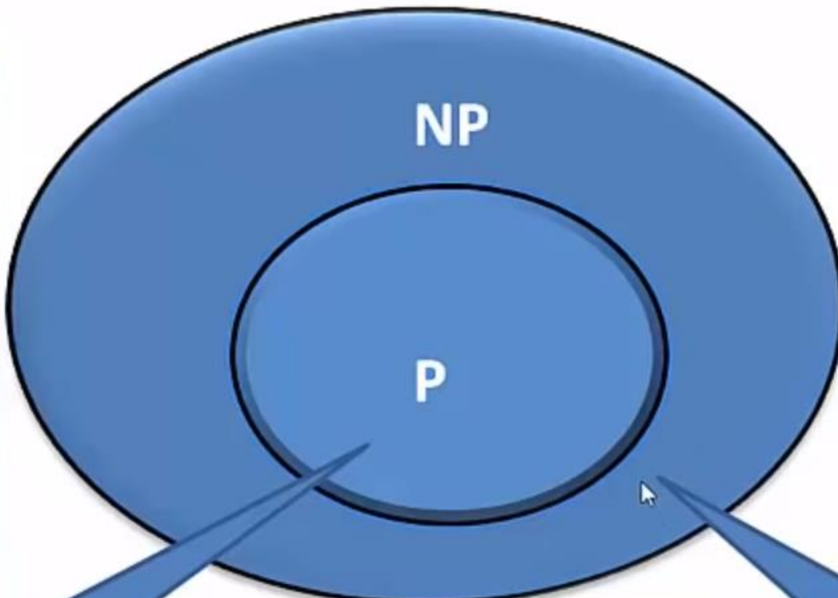
<u>NP Class</u>

Hard to Solve

&

Easy to Verify

Exponential time



<u>P Class</u>

Easy to Solve

&

Easy to Verify

Polynomial time

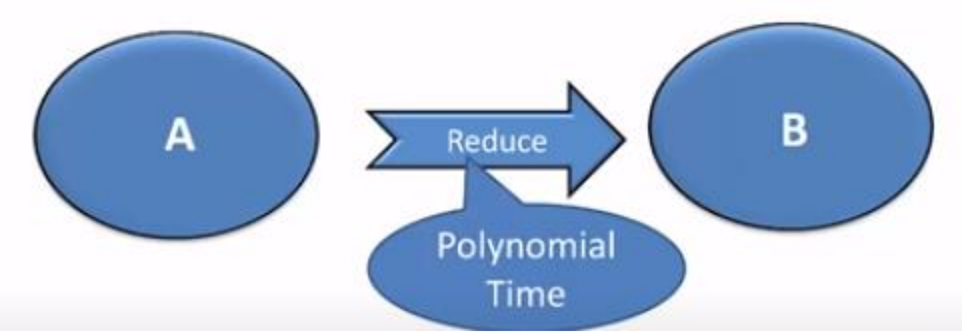
Tractable

$P \subseteq NP$

Intractable

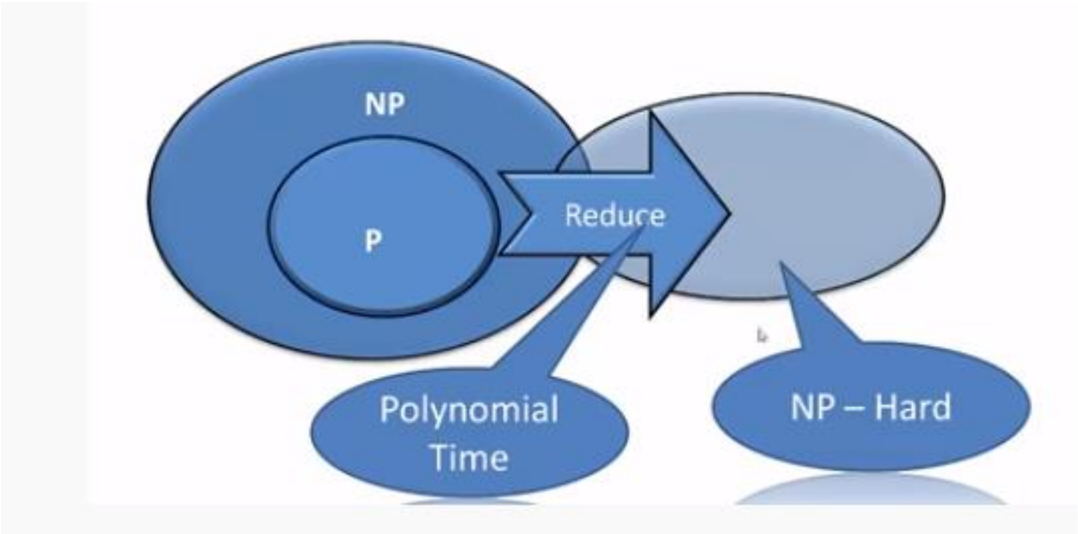
Reduction

- We have two problems, A and B, and we know problem B is a P class problem. If problem A can be reduced, or converted to problem B, and this reduction takes a polynomial amount of time, then we can say that A is also a P class problem (A is reducible to B).



NP-Hard Problems

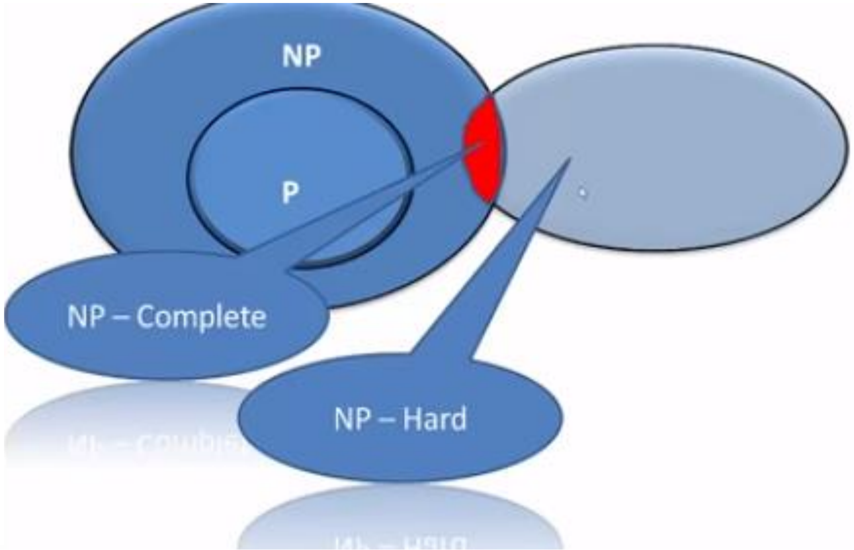
- A problem is classified as NP-Hard when an algorithm for solving it can be translated to solve *any* NP problem. Then we can say, this problem is *at least* as hard as any NP problem, but it could be much harder or more complex.
- **NP-hard**-- Now suppose we found that A is reducible to B, then it means that B is at least as hard as A.



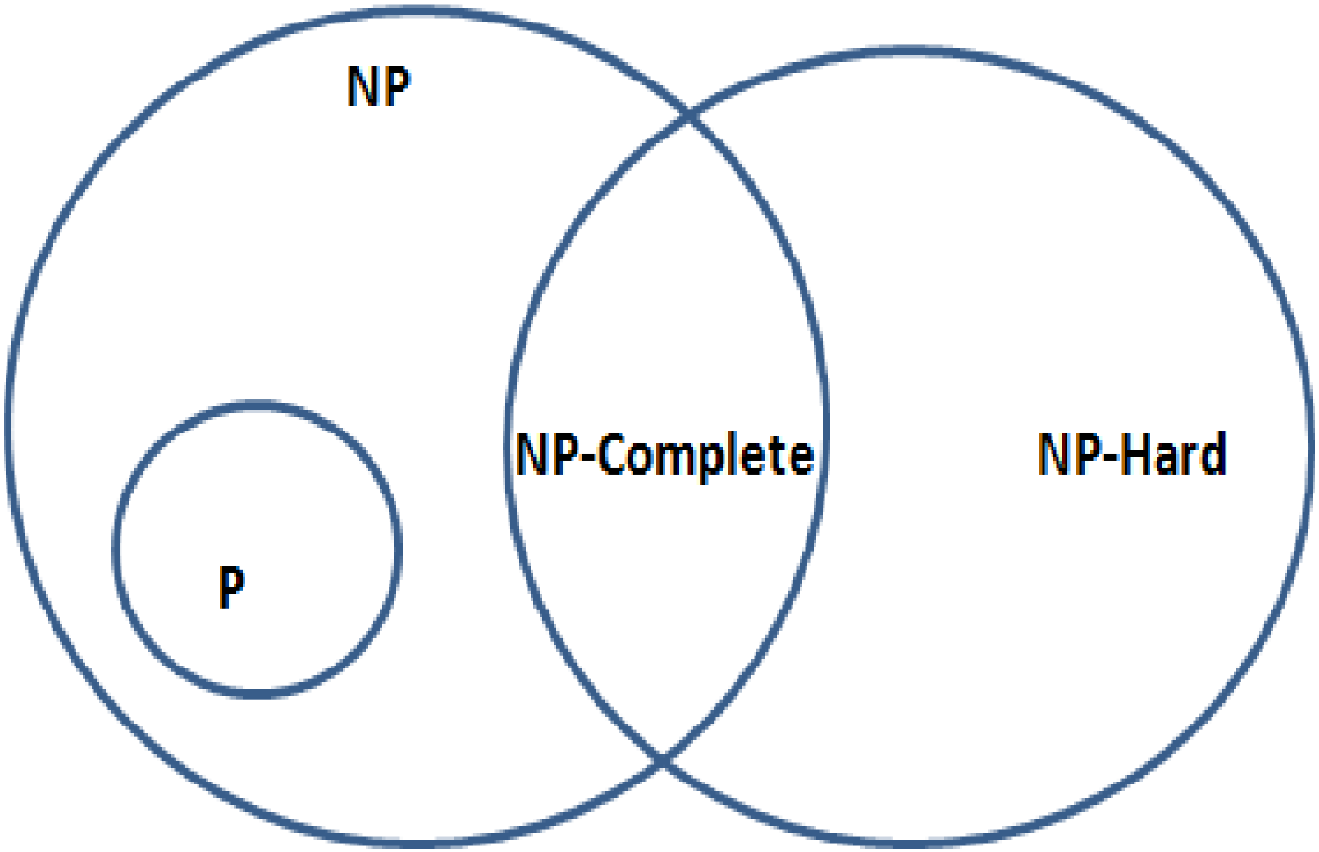
NP-Complete Problems

- NP-Complete problems are problems that live in both the NP and NP-Hard classes. This means that NP-Complete problems can be verified in polynomial time and that any NP problem can be reduced to this problem in polynomial time.

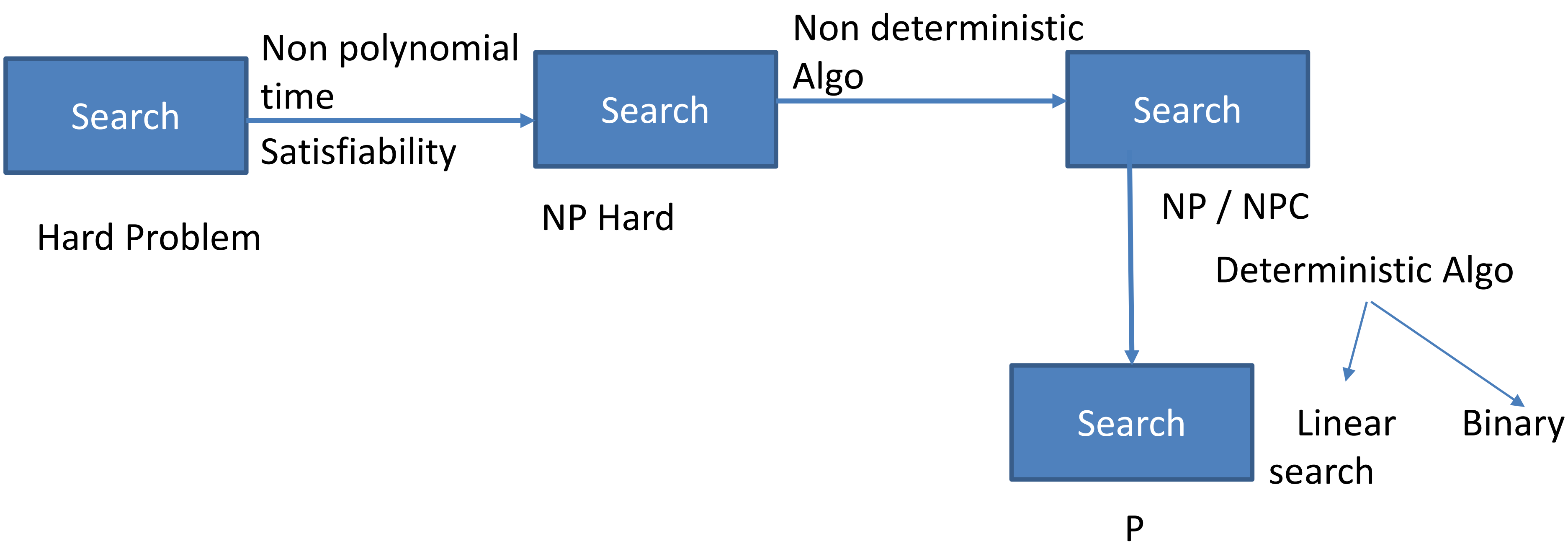
- The group of problems which are both in NP and NP-hard are known as NP-Complete problem.
- Now suppose we have a NP-Complete problem R and it is reducible to Q then Q is at least as hard as R and since R is an NP-hard problem. therefore Q will also be at least NP-hard , it may be NP-complete also.



Venn diagram



Example


Search
Non polynomial time
Satisfiability
Hard Problem
Search
NP Hard
Non deterministic Algo
Search
NP / NPC
Deterministic Algo
Search
Linear search
Binary
P

Lecture 33

NP-Completeness

NP-complete problems

- A decision problem *D is NP-complete iff*

  1. D $\in$ *NP*

  2. every problem in *NP is polynomial-time reducible to D*

- Other *NP-complete problems obtained through* polynomial-time reductions of known *NP-complete* problems

- **Reduction**

- A problem P can be *reduced to another problem Q if* any instance of P can be rephrased to an instance of Q, the solution to which provides a solution to the instance of P
  - » This rephrasing is called a *transformation*
- Intuitively: If P reduces in polynomial time to Q, P is “no harder to solve” than Q

Polynomial-time reductions

- **Informal explanation of reductions:**
- We have two problems, X and Y. Suppose we have a black-box solving problem X in polynomial-time. Can we use the black-box to solve Y in polynomial-time?
- If yes, we write $\mathbf{Y \leq_P X}$ and say that Y is **polynomial-time reducible to** X.
- More precisely, we take any input of Y and in polynomial number of steps translate it into an input (or a set of inputs) of X. Then we call the black-box for each of these inputs. Finally, using a polynomial number of steps we process the output information from the boxes to output the answer to problem Y.

NP-complete and NP-hard: how to prove

- **The recipe to prove NP-hardness of a problem X:**

1. Find an already known NP-hard problem Y.
2. Show that Y $\leq_P$ X.

- **The recipe to prove NP-completeness of a problem X:**

1. Show that Y is NP-hard.
2. Show that Y is in NP.

Proving NP-Completeness

What steps do we have to take to prove a problem Q is NP-Complete?

» Pick a known NP-Complete problem P

» Reduce P to Q

- Describe a transformation that maps instances of P to instances of Q, s.t. “yes” for Q = “yes” for P
- Prove the transformation works
- Prove it runs in polynomial time

» Prove Q ∈ **NP**

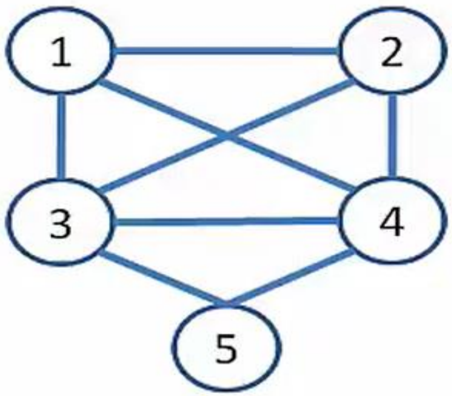
Proving Vertex Cover Problem to NP Complete Problem

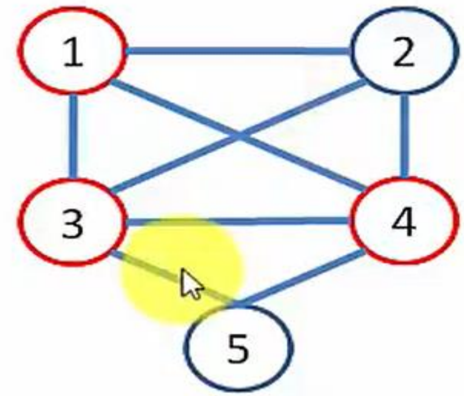
Vertex Cover Problem

A vertex cover of a graph is a set of vertices that touches every edge in the graph.

A vertex cover of a graph $G = (V, E)$ is a subset $Vc \subseteq V$ such that if $(a, b) \in E$ then either $a \in V_C$ OR $b \in V_C$ OR both.

Vertex Cover Problem





A vertex cover of a graph G = (V, E) is a subset Vc ⊆ V such that if (a, b) ∈ E then either a ∈ Vc OR b ∈ Vc OR both a, b ∈ Vc.

Steps for proving NP-Complete:

Step 1: Prove that B is in NP

Step 2: Select an NP-Complete Language A.

Step 3: Construct a function f that maps members of A to members of B.

Step 4: Show that x is in A iff f(x) is in B.

Step 5: Show that f can be computed in polynomial time.

NP-Completeness of Vertex Cover Problem:

Vertex Cover Problem is in NP.

Vertex Cover Problem is NP- Hard.

NP-Completeness of Vertex Cover Problem:

To Show Vertex Cover Problem is in NP.

A Problem which cannot be solved on polynomial time but is verified in polynomial time is known as Non Deterministic Polynomial or NP-Class Problem.

Given Vc, vertex cover of $G = (V, E)$, $|Vc| = k$. We can check in $O(|V| + |E|)$ that Vc is a vertex cover for G.

For each vertex $\in$ Vc , remove all incident edges. Check if all edges were removed from G.

Thus Vertex Cover Problem is in NP.

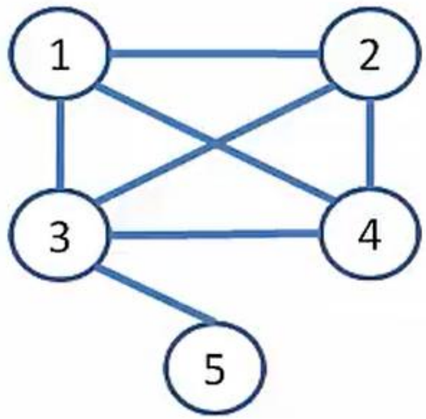
NP-Completeness of Vertex Cover Problem:

To show Vertex Cover Problem is NP- Hard.

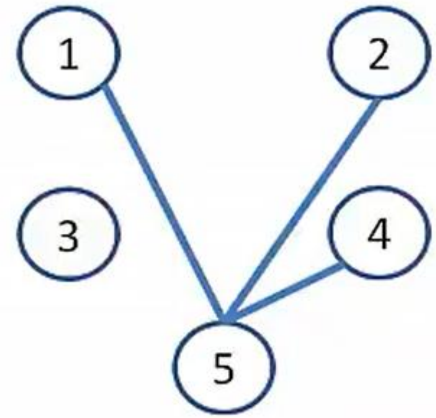
we need to show that Vertex Cover is at least as hard any other problem in NP.

we give a reduction from Clique to Vertex Cover Problem.

It means, given an instance *I* of Clique, we will produce a graph G(V,E) and an integer k such that G has a maximum clique of k if and only if *I* in $\bar{G}(V,\bar{E})$ has a vertex cover of size |V|-k.

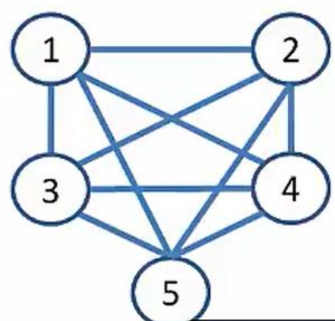


$G(V,E)$



$\bar{G}(V,\bar{E})$

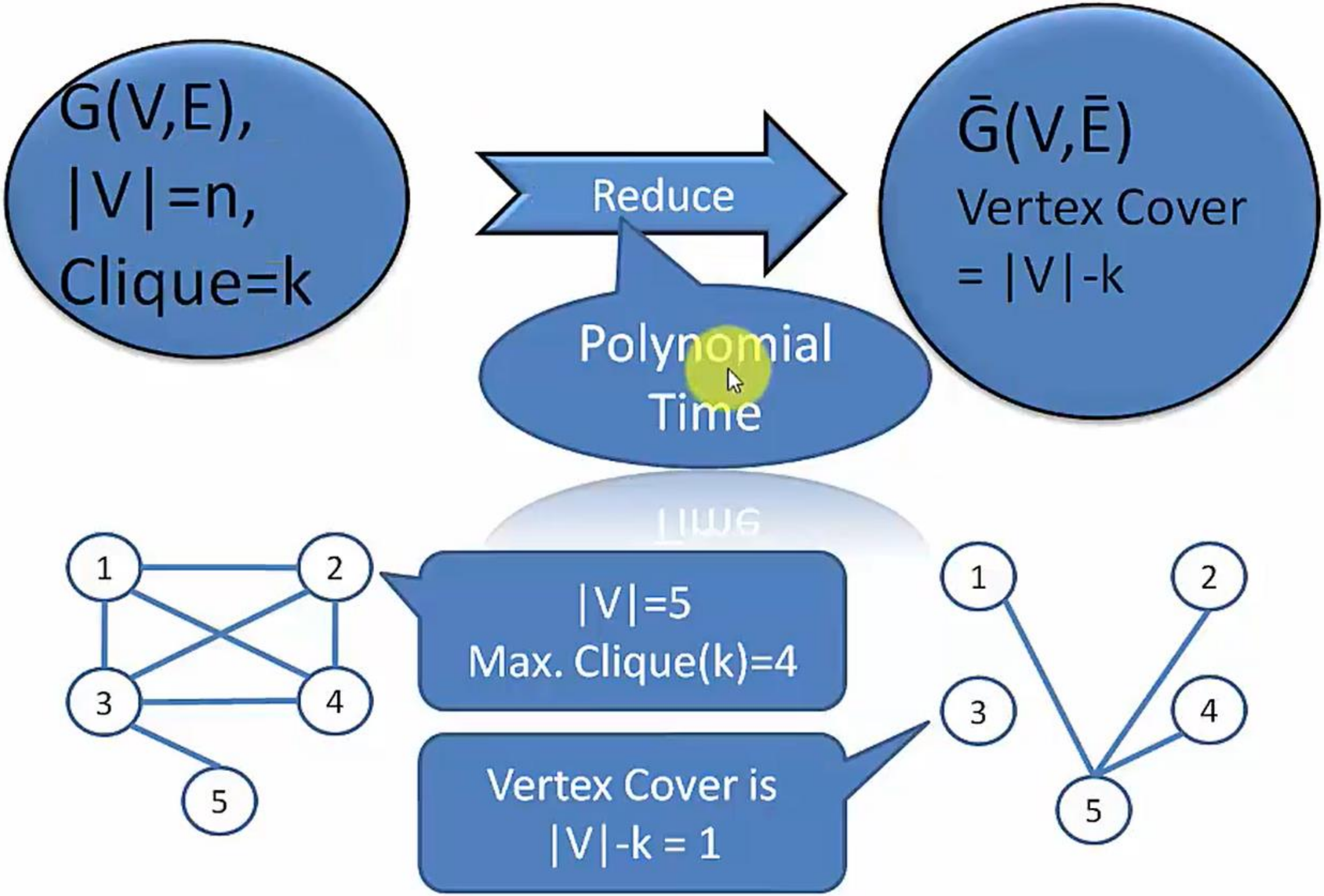
$G(V,E)$ $\bar{G}(V,\bar{E})$



Complement $\bar{G}(V,\bar{E})$ of a graph $G(V,E)$ is a graph such that $\bar{E} = \{(u,v) \in G \mid (u,v) \notin \bar{G}\}$

OR

$G(V,E) \cup \bar{G}(V,\bar{E})$ is a complete graph



INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Let G has clique V' of size k.

=> G has vertex cover of size $|V|-k$

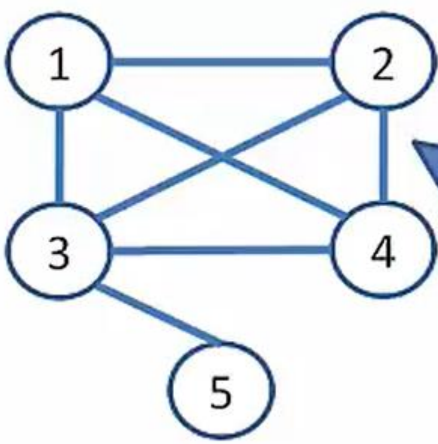
$(a, b) \in E \Rightarrow (a, b) \notin \bar{E}$

If $(a, b) \in \bar{E}$, then at least a or b $\notin V'$.

Every pair in V' is connected by an edge in E.

=> At least one of a or b is in V-V'

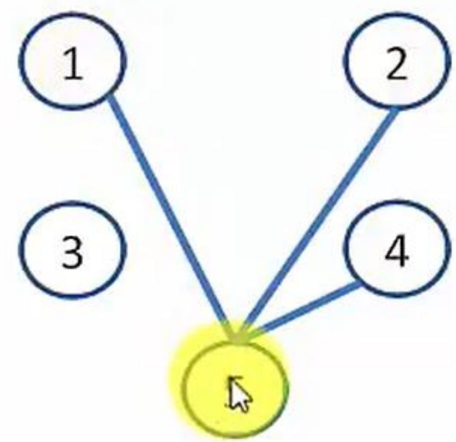
=> Edge (a,b) is covered by V-V'



$V=\{1,2,3,4,5\}$

$V'=\{1,2,3,4\}$

$V-V'=\{5\}$



- Thus, we can say that there is a clique of size k in graph G if and only if there is a vertex cover of size |V| – k in G’, and hence, any instance of the clique problem can be reduced to an instance of the vertex cover problem. Thus, vertex cover is NP Hard. Since vertex cover is in both NP and NP Hard classes, it is NP Complete.

https://www.geeksforgeeks.org/proof-that-vertex-cover-is-np-complete/